1. the difference between internal and external linkage using examples from your code

Internal linkage means that it is only visible in the current translation unit. On the other hand, external linkage means that it is visible for the whole program.

By default, the compiler will treat the non-const global variable and functions with the external linkage. In addition, the const global variables with the internal linkage by default.

For example, in my string.cpp file, the 'extern const int INITIAL'. Because of the “extern” keyword, the compiler will try to find the INITIAL declared in another translation unit. However, if we replace by the static keyword, such as 'static int INITIAL', then the new INITIAL will only be visible in this translation unit. Other translation units will not be able to use this variable.

1. the execution effect of the code in the **process()** function that creates the **String** object which is inserted into standard output (which member function(s) does this code call)

By create the String object to hold the unmodifiable c-style null terminated string, we can define the way it inserted into standard output. In this case, we add the counter number at the beginning and then control the length of the string. However, if we just simply insert the c-style string into the standard output, it will only display the string without any modification.

1. the changes that you made in upgrading your **String** class.

In order to let the program to handle any length of the c-style null terminated string, I add the char type pointer to hold the address of the string and the integer variable for the length of the string. By using dynamic memory allocation, we don’t need to determine the size of the char array. Once the user provides the string, the program will automatically count the length and allocate the memory for it. In addition, because of the dynamic memory allocation, I add the destructor to deallocate the dynamic memory.